Errata Sheet

Rel. 1.2, 2012-07-17

Device **TC1798/93/91, TC1748**

Marking/Step **ES-AB, AB**

Package **see Data Sheet**

02698AERRA

This Errata Sheet describes the deviations from the current user documentation.

Table 1 Current Documentation[1)]

TC1798 User's Manual	V1.2	2012-05
TC1798 Data Sheet	V1.0	2012-03
TC1793 Data Sheet	V1.1.1	2012-12
TC1791 Data Sheet	V1.0	2012-01
TriCore V1.6 Core Architecture, Instruction Set	V1.0	2010-03

1) Newer versions replace older versions, unless specifically noted otherwise.

Make sure you always use the corresponding documentation for this device (User's Manual, Data Sheet, Documentation Addendum (if applicable), TriCore Architecture Manual, Errata Sheet) available in category 'Documents' at **www.infineon.com/AudoMax** and **www.myInfineon.com**.

Conventions used in this document

Each erratum identifier follows the pattern **Module_Arch.TypeNumber**:

- **Module**: subsystem, peripheral, or function affected by the erratum
- **Arch**: microcontroller architecture where the erratum was initially detected
  - **AI**: Architecture Independent
  - **TC**: TriCore
- **Type**: category of deviation
  - **[none]**: Functional Deviation
  - **P**: Parametric Deviation

  - **H**: Application Hint
  - **D**: Documentation Update
- **Number**: ascending sequential number within the three previous fields. As this sequence is used over several derivatives, including already solved deviations, gaps inside this enumeration can occur.

Notes

1. This Errata Sheet applies to all temperature and frequency versions and to all memory size variants, unless explicitly noted otherwise. For a derivative synopsis, see the latest Data Sheet/User's Manual.
   This Errata Sheet covers several device versions. If an issue is related to a particular module, and this module is not specified for a specific device version, this issue does not apply to this device version.
   E.g. issues with identifier "EBU" do not apply to TC1791 devices where no EBU is specified.
2. Devices marked with EES or ES are engineering samples which may not be completely tested in all functional and electrical characteristics, therefore they should be used for evaluation only.
   The specific test conditions for EES and ES are documented in a separate Status Sheet.
3. This device is equipped with a TriCore "TC1.6" core. Some of the errata have workarounds which are possibly supported by the tool vendors. Some corresponding compiler switches need possibly to be set. Please see the respective documentation of your compiler.
   For effects of issues related to the on-chip debug system, see also the documentation of the debug tool vendor.

1 History List / Change Summary

Table 2 History List

Version	Date	Remark
1.0	2011-07-20	
1.1	2012-04-17	ADC_TC.061 renamed to ADC_AI.H003
1.2	2013-07-17	CPU_TC.H007 replaced by CPU_TC.122

Note: Changes to the previous errata sheet version are particularly marked in column “Change” in the following tables.

Table 3 Errata fixed in this step

Errata	Short Description	Change
DMI_TC.018	SPB Access to DSPR at D800 8000 $_H$.. D801 FFFF $_H$	Fixed
OCDS_TC.029	Cerberus read/write access to OVC registers	Fixed

Table 4 Functional Deviations

Functional Deviation	Short Description	Change	Page
BROM_TC.006	**Baud Rate Detection for CAN Bootstrap Loader**		**10**
CPU_TC.117	**Cached Store Data Lost on Data Cache Invalidate via Overlay**		**10**
CPU_TC.118	**Atomic operations targeting TriCore DSPR fail to lock memory.**		**12**

Table 4 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
CPU_TC.119	Return Address may be set to 0x0000_0000 after an FCD trap		13
CPU_TC.120	A load using circular addressing may return incorrect data		13
CPU_TC.122	Speculative instruction fetches may trigger unexpected error exceptions	New	14
EBU_TC.023	Read Data Corruption when accessing SDRAM/DDRAM		20
FADC_TC.005	Equidistant multiple channel-timers		21
FlexRay_AI.087	After reception of a valid sync frame followed by a valid non-sync frame in the same static slot the received sync frame may be ignored		22
FlexRay_AI.088	A sequence of received WUS may generate redundant SIR.WUPA/B events		23
FlexRay_AI.089	Rate correction set to zero in case of SyncCalcResult=MISSING_TERM		24
FlexRay_AI.090	Flag SFS.MRCS is set erroneously although at least one valid sync frame pair is received		25
FlexRay_AI.091	Incorrect rate and/or offset correction value if second Secondary Time Reference Point (STRP) coincides with the action point after detection of a valid frame		25
FlexRay_AI.092	Initial rate correction value of an integrating node is zero if pMicroInitialOffsetA,B = 0x00		26
FlexRay_AI.093	Acceptance of startup frames received after reception of more than gSyncNodeMax sync frames		27

Table 4 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
FlexRay_AI.094	Sync frame overflow flag EIR.SFO may be set if slot counter is greater than 1024		27
FlexRay_AI.095	Register RCV displays wrong value		28
FlexRay_AI.096	Noise following a dynamic frame that delays idle detection may fail to stop slot		29
FlexRay_AI.097	Loop back mode operates only at 10 MBit/s		30
FlexRay_AI.099	Erroneous cycle offset during startup after abort of startup or normal operation		30
FlexRay_AI.100	First WUS following received valid WUP may be ignored		31
FlexRay_AI.101	READY command accepted in READY state		32
FlexRay_AI.102	Slot Status vPOC!SlotMode is reset immediately when entering HALT state		32
FlexRay_AI.103	Received messages not stored in Message RAM when in Loop Back Mode	New	33
GPT12_TC.001	T2IN/T3IN Port Connections		33
MSC_TC.009	Missing Receive Data Interrupt		34
OCDS_TC.035	Debug reset will not disable the OCDS		34
PCP_TC.023	JUMP sometimes takes an extra cycle		35
PCP_TC.032	Incorrect PCP behaviour following FPI timeouts (as a slave)		35
PCP_TC.041	Coincident PCP non-atomic Write and FPI RMW Access to PRAM		36
SRI_TC.001	Atomic CPU operation with Bus Address ECC Error will cause bus hang		37
SRI_TC.002	SRI_XBAR Debug registers may capture wrong transaction during pipelining		37

Table 4 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
SSC_AI.022	Phase error detection switched off too early at the end of a transmission		38
SSC_AI.023	Clock phase control causes failing data transmission in slave mode		38
SSC_AI.024	SLSO output gets stuck if a reconfig from slave to master mode happens		38
SSC_AI.025	First shift clock period will be one PLL clock too short because not syncronized to baudrate		39
SSC_AI.026	Master with highest baud rate set generates erroneous phase error		39

Table 5 Deviations from Electrical- and Timing Specification

AC/DC/ADC Deviation	Short Description	Change	Page
AC_PER_TC.P001	Driver Settings for SSC and MLI Outputs on A2 Pads		41
PLL_ERAY_TC.P001	PLL lock-in time		41
PLL_TC.P006	PLL lock-in time		41

Table 6 Application Hints

Hint	Short Description	Change	Page
ADC_AI.H002	Minimizing Power Consumption of an ADC Module		42
ADC_AI.H003	Injected conversion may be performed with sample time of aborted conversion		42
BROM_TC.H002	Enabling CAN Communication in CAN Bootstrap Loader Mode		44
CPU_TC.H005	Wake-up from Idle/Sleep Mode		44
CPU_TC.H006	Store Buffering in TC1.6/P/E Processors		45
EBU_TC.H010	EBU SFR access for Non-TriCore masters after reset		47
EBU_TC.H011	Hints for DDR PCB Layout		47
FlexRay_AI.H004	Only the first message can be received in External Loop Back mode		48
FlexRay_AI.H005	Initialization of internal RAMs requires one eray_bclk cycle more		49
FlexRay_AI.H006	Transmission in ATM/Loopback mode		49
FlexRay_AI.H007	Reporting of coding errors via TEST1.CERA/B		49
FlexRay_AI.H009	Return from test mode operation		49
FlexRay_AI.H010	Driver SW must launch CLEAR_RAMS command before reading from E-Ray RAMs		50
GPT12_AI.H001	Modification of Block Prescalers BPS1 and BPS2		50
GPTA_TC.H004	Handling of GPTA Service Requests		51
INT_TC.H003	Safe Cancellation of Service Requests	New	55
LMU_TC.H001	Behaviour of Error Flags in Register LMU_MEMCON		55
MSC_TC.H007	Start Condition for Upstream Channel		55

Table 6 Application Hints (cont'd)

Hint	Short Description	Change	Page
MSC_TC.H010	Configuration of SCU.EMSR for the EMGSTOPMSC Signal	New	56
MultiCAN_AI.H005	TxD Pulse upon short disable request		57
MultiCAN_AI.H006	Time stamp influenced by resynchronization		57
MultiCAN_AI.H007	Alert Interrupt Behavior in case of Bus-Off		57
MultiCAN_AI.H008	Effect of CANDIS on SUSACK		58
MultiCAN_TC.H002	Double Synchronization of receive input		58
MultiCAN_TC.H003	Message may be discarded before transmission in STT mode		59
MultiCAN_TC.H004	Double remote request		59
OCDS_TC.H001	IOADDR may increment after aborted IO_READ_BLOCK		60
OCDS_TC.H002	Setting IOSR.CRSYNC during Application Reset		60
OCDS_TC.H003	Application Reset during host communication		61
OCDS_TC.H007	Early Acknowledgement of Channel Suspend for Active DMA Channel		62
SCU_TC.H004	Flag TRAPSTAT.SYSVCOLCKT after Power-on or System Reset		62
SENT_TC.H001	Functionality of RCRx.CRZ: Differences to 2010 Standard		63
SSC_AI.H002	Transmit Buffer Update in Master Mode during Trailing or Inactive Delay Phase		63
SSC_AI.H003	Transmit Buffer Update in Slave Mode during Transmission		64

Table 6 Application Hints (cont'd)

Hint	Short Description	Change	Page
SSC_TC.H003	Handling of Flag STAT.BSY in Master Mode		64

Table 7 Documentation Updates

Hint	Short Description	Change	Page
EBU_TC.D001	Documentation Updates related to SRI External Bus Unit (EBU)	New	66
SSC_TC.D001	Documentation Updates related to SSC MRST/MTSR Master/Slave Attribute	New	67

2 Functional Deviations

BROM_TC.006 Baud Rate Detection for CAN Bootstrap Loader

In a specific corner case, the baud rate detected during reception of the initialization frame for the CAN bootstrap loader may be incorrect. The probability for this sporadic problem is relatively low, and it decreases with decreasing CAN baud rate and increasing module clock frequency.

Workaround:

If communication fails, the host should repeat the CAN bootstrap loader initialization procedure after a reset of the device.

CPU_TC.117 Cached Store Data Lost on Data Cache Invalidate via Overlay

Cached store data can be lost if the overlay system requests a data cache invalidate in the same cycle as a cache line is being written. The overlay control provides a mechanism to do a single cycle invalidate of all valid/clean lines in the data cache by writing the OCON.DCINVAL bit. Please note that there is no problem if the data cache is used exclusively for read data (e.g. flash constants).

```
Cache line state transition on DCINVAL.
   valid/clean -> (DCINVAL) -> invalid/clean
```

A normal store operation transitions the cache line to a valid/dirty state.

```
Cache line state transition on normal store operation.
    valid/clean -> (write) -> valid/dirty
    invalid/clean -> (write) -> valid/dirty
```

In the case where the write and invalidate are received in the same cycle, the dirty bit is correctly updated but the valid bit is incorrectly cleared.

```
Cache line state transition on store operation with DCINVAL
     valid/clean -> (write+DCINVAL)- > invalid/dirty
```

```
invalid/clean -> (write+DCINVAL) -> invalid/dirty
```

This leads to a loss of data as the store data ends up being held in an invalid cache line and hence never re-read.

Workaround-1

Ensure that the data cache is never used to cache write data. This can be ensured by software design but may limit performance in some systems.

Workaround-2

Ensure that the core is never storing data when OCON.DCINVAL is asserted.

This requires the CPUs store buffers to be empty when the invalidate is asserted. This can only be done by getting the CPU to firstly flush all write data with a DSYNC command, then to write the OCON.DCINVAL to trigger an invalidate.

The following example code sequence performs the required operations:-

- Read the OCON register to get the current SHOVEN field
- Create a new OCON value with DCINVAL, OVSTRT and OVCONF bits set
- Perform a DSYNC operation to flush all write data to memory
- Write OCON with the new value.
- Read back OCON to ensure write is complete

```
;; Set up A14 with address of OCON Register
movh.a  a14,#(((0xF87FFBE0)+0x8000>>16) & 0xffff)lea
a14,[a14]((((0xF87FFBE0)+0x8000)&0xffff)-0x8000)
;; Load a15 with contents of OCON
ld.w   d15, [a14]
;; Set OCONF, DCINVAL, OVSTRT start values
movh   d14  , #0x0305
;; Combine existing SHOVEN
insert d15, d14,d15,#0,#16
; Flush all store data
dsync
```

```
;; Store New value back to OCON
st.w    [a14], d15
;; Re-read to ensure store is complete
ld.w    d15, [a14]
```

Attention: This routine must be run with interrupts disabled, either as part of an interrupt service routine or guarded by enable/disable instructions.

This routine may be run periodically or run as part of a dedicated interrupt service routine. If the latter approach is used it is suggested that an unused SRN either in the CPU or Cerberus is utilised to trigger the invalidate. In all cases the routine must be run with interrupts disabled to ensure that no writes are in progress when the invalidate occurs.

The OCON.OVCONF bit may be used to indicate the state of the invalidate operation. If it is cleared in advance, the routine above will set it when the cache invalidate operation is performed.

CPU_TC.118 Atomic operations targeting TriCore DSPR fail to lock memory.

The TriCore TC1.6 instruction set includes the following atomic instructions

- SWAP.W
- ST.T
- LDMST

These instructions each perform load-modify-store type operations on a target memory location. To ensure the atomicity of the operation, the target memory location is required to be “locked” between the load and the store to prevent any other memory transaction occurring.The lock functionality of the TriCore DSPR memory cannot be guaranteed, hence the atomicity of operations targeting DSPR memory cannot be guaranteed.

Typically atomic instructions are used to build semaphores, allowing the synchronisation of tasks running on separate processors.

Workaround

As this problem does not affect any of the other memories in the system the target of all atomic instructions should be placed in LMU or PSPR memory.

CPU_TC.119 Return Address may be set to 0x0000_0000 after an FCD trap

A problem exists whereby the return address (A11) is not correctly set after taking an FCD (Free Context List Depleted) trap, and hence leading to an unrecoverable state. The problem occurs when an asynchronous trap request or an interrupt is received by the TriCore CPU while an FCD trap is pending to be taken. In this case the trap entry sequence begins and the upper context registers are stored to the appropriate CSA (Context Save Area). However, the return address is incorrectly set to 0x0000_0000.

Workaround

Treat FCD as unrecoverable if the return address (A11) is equal to 0x0000_0000 in the FCD handler.

CPU_TC.120 A load using circular addressing may return incorrect data

Under the following circumstances a load using the circular addressing mode may return incorrect data.

- The circular buffer base is located at an offset of 8,18,28,etc.
  - Base address mod 0x10 == 0x8
- The load wraps from one end of the buffer to the other (Index + Access_Size > Length)
  - This causes the access to be split into two, The first from the top of the buffer, the second from the base.
- The load is immediately preceded by a store to a location at the buffer base such that it will be the target of the second load.

Under the above conditions the second load fails to correctly recognise the store and returns the memory data prior to the store.

Workarounds

Any of the following workarounds will resolve this issue

- Locate the buffer base address at a 128 Bit alignment (0,10,20,30 etc)
- Ensure that no load access wraps around the end of the buffer.
- Ensure that a store is not performed immediately before a circular load
- Insert a NOP between the store and load

Note: The TriCore circular addressing mode cannot be compiler generated, hence this errata only effects hand generated assembler code.

CPU_TC.122 Speculative instruction fetches may trigger unexpected error exceptions

To improve performance the TriCore TC1.6 processors prefetch instructions in many situations. In general the prefetch address will be a valid program location as dictated by the instruction flow.

In the case of return instructions (RET, RFE, FRET etc.) the processor will attempt to predict the return address and prefetch the return target instruction. In most situations this prediction is extremely easy and 100% accurate. The return target is the value in A11. Under some circumstances the prediction mechanism for return instructions fails. When return prediction fails we see erroneous prefetch addresses being generated. If these prefetch addresses are outside of a valid memory location unexpected error exceptions (interrupts and NMIs) may be generated.

Note: The TriCore TC1.6 processor will only ever execute code as dictated by the instruction flow. Prefetches from invalid locations are always discarded and never executed.

The TriCore TC1.6 processor will only ever prefetch instructions from locations for which it has execute permissions:

- If the MPU is enabled these are the regions defined by the memory protection range settings.
- If the MPU is disabled these are memory segments 0 to 14.
- Instructions can never be prefetched from the peripheral segment (Segment 15).

Although prefetching from invalid locations has no effect on the execution of code on the CPU it can cause undesirable side effects within the overall system. Such side effects include unexpected error interrupts from the bus and memory systems, and ECC error interrupts/NMI.

Recommendation

To mitigate these side effects we recommend enabling the memory protection (MPU) system.

If enabling the MPU is not possible then the following mitigation measures should be implemented to avoid these side effects:

- Initialise all physical memory in the system.[1)]
- Do not place ECC errors in memory.
- Disable SRI XBAR unimplemented address interrupt.
- Disable PMU protection error interrupt.
- Disable SRI XBAR read error interrupt for the PMU.
- Disable SRI XBAR read error interrupt for the EBU.
- Do not use read sensitive devices on the EBU.

MPU programming example

To prevent erroneous prefetching a simple static configuration of the MPU may be used. Such a configuration defines the allowable executable memory space for the application and is used as a filter for fetch addresses. If the MPU is used only to prevent unexpected fetches no MPU trap handlers are required. When the MPU is enabled, only accesses that match in the protection ranges are allowed to leave the CPU. Configuring the MPU code ranges to cover only the used (or valid) memory areas prevents all potential side effects from speculative fetch.

1) All physical memory must be initialised including the unusable/unavailable memory in derivative devices. For devices using address ECC this must include the full 4MBytes of PFlash. If address ECC is not used only the logical PFlash memory area needs to be initialised (3MBytes for TC1791*-384F*, 4MBytes for TC1798/3/1*-512* devices).

Usually code is executed from Program Flash (PFlash) or from Program Scratch Pad RAM (PSPR).

In TC1798, TC1793 and TC1791*-512F* devices with **4MB PFlash**, this corresponds to the following address ranges:

- 0x8000_0000 – 0x801F_FFFF – PFlash0 (2MB)
- 0x8080_0000 – 0x809F_FFFF – PFlash1 (2MB)
- 0xC000_0000 – 0xC000_7FFF – PSPR (32KB)

In TC1791*-384F* devices with **3MB PFlash**, this is:

- 0x8000_0000 – 0x801F_FFFF – PFlash0 (2MB)
- 0x8080_0000 – 0x808F_FFFF – PFlash1 (1MB)
- 0xC000_0000 – 0xC000_7FFF – PSPR (32KB)

Uninitialised memory ranges should be excluded. Valid (initialised) memory ranges that are not used by code may be included if convenient. For example, with the following address map:

- 0x8000_0000 – 0x800C_F7FF – code 1
- 0x800C_F800 – 0x800F_FFFF – not used, initialised
- 0x8010_0000 – 0x8017_FFFF – data1
- 0x8018_0000 – 0x8019_FFFF – code2
- 0x801A_0000 – 0x801F_FFFF – not used, not initialised
- 0x8080_0000 – 0x8088_FFFF – data2
- 0x8089_0000 – 0x808F_FFFF – not used, not initialised

It is possible to define two code ranges for protection:

- code range A: 0x8000_0000 – 0x800C_F7FF
- code range B: 0x8018_0000 – 0x8019_FFFF

Or one code range (because the area between code ranges has valid data):

- code range A: 0x8000_0000 – 0x8019_FFFF

TriCore implementation in TC1798, TC1791, TC1793 can handle up to 4 protection code ranges. An example configuration is shown below.

```
// clear COMPAT.PROT bit to enable TC1.6 extensions to  the MPU
// Note this register is ENDINIT protected
__mtcr(COMPAT, __mfcr(COMPAT) & ~(1<<2));
__isync();
```

```
// allow code (fetch) access to the specified address ranges
__mtcr(CPS0, 0x8888);     // xe[0..3] = 1, rs[0..3] = 0
__mtcr(CPS1, 0x8888);     // xe[0..3] = 1, rs[0..3] = 0
__mtcr(CPS2, 0x8888);     // xe[0..3] = 1, rs[0..3] = 0
__mtcr(CPS3, 0x8888);     // xe[0..3] = 1, rs[0..3] = 0
__mtcr(CPR0_0L, code_rangeA.start);
__mtcr(CPR0_0U, code_rangeA.end+1);
__mtcr(CPR1_0L, code_rangeB.start);
__mtcr(CPR1_0U, code_rangeB.end+1);
__mtcr(CPR2_0L, code_rangeC.start);
__mtcr(CPR2_0U, code_rangeC.end+1);
__mtcr(CPR3_0L, code_rangeD.start);
__mtcr(CPR3_0U, code_rangeD.end+1);
// allow data access to the whole address space
__mtcr(DPS0, 0xC);     // we0=1, re0=1, rs0=0
__mtcr(DPS1, 0xC);     // we0=1, re0=1, rs0=0
__mtcr(DPS2, 0xC);     // we0=1, re0=1, rs0=0
__mtcr(DPS3, 0xC);     // we0=1, re0=1, rs0=0
__mtcr(DPR0_0L,  0x00000000);
__mtcr(DPR0_0U, 0xFFFFFFFF);
__isync();
// set SYSCON.PROTEN to activate MPU function
__mtcr(SYSCON, __mfcr(SYSCON) | (1<<1));
__isync();
```

If the application has not been using MPU before it is likely that PSW.PRS field is not used and is never modified (leaving it with the reset value of zero). If PSW.PRS field value is always zero, the MPU configuration can be simplified by removing writes to CPS1, CPS2, CPS3, DPS1, DPS2, DPS3 registers.

Potential side effects and mitigations if enabling the MPU is not possible

1. Prefetch to unimplemented memory space:

If a prefetch accesses an unimplemented/reserved memory space the following side effects will occur:

- An error interrupt will be signalled by the SRI Xbar bus system and the Xbar error register updated.

To mitigate against this either the Xbar unimplemented address interrupt generation can be disabled or the overall Xbar interrupt can be disabled

- XBAR_ARBCOND.PRERREN = 0_B (default after reset: 1_B)
- XBAR_SRC.SRE = 0_B

2. Prefetch to uninitialized memory or memory containing ECC error(s) for test purposes

If a prefetch accesses an uninitialized location or a memory location that contains an ECC error the following side effects will occur:

- An error will be flagged by the memory system to the SCU (LMU, PSPR, DSPR). This will typically result in an NMI.
- An error interrupt will be signalled (Flash memory).
- An interrupt will be signalled by the SRI Xbar bus system and the Xbar error register updated.

To mitigate against this either the memory should be fully initialised and ECC errors not inserted or the ECC error interrupt generation for the memory should be disabled.

To disable ECC error interrupts the following SCU configuration bits should be cleared

- SCU_ECCCON.ECCENLMU = 0_B (default after reset: 1_B)
- SCU_ECCCON.ECCENSRI = 0_B (default after reset: 1_B)
- SCU_ECCCON.ECCENDSPR = 0_B (default after reset: 1_B)
- SCU_ECCCON.ECCENPSPR = 0_B (default after reset: 1_B)

Additionally for the Flash memories the following ECC interrupt masks should be disabled in registers PMUx_FCON (x = 0, 1)

- PMUx_FCON.PFDBERM = 0_B

- PMUx_FCON.PFSBERM = 0_B
- PMUx_FCON.DFCBERM = 0_B
- PMUx_FCON.DFMBERM = 0_B

3. Prefetch to unreadable memory location in flash memory

Flash memory locations may be read protected (reserved by SHE) or busy due to programming, erase etc. If a prefetch attempts to access such a location the following side effects will occur:

- An error interrupt will be signalled by the PMU.
- An interrupt will be signalled by the SRI Xbar bus system and the Xbar error register updated.

To mitigate against this the flash protection error interrupt generation can be disabled and the Xbar interrupt generation can be disabled

- PMU_FCON.PROERM = 0_B
- XBAR_ARBCON3,4.PRERREN = 0_B (default after reset: 1_B) or XBAR_SRC.SRE = 0_B

4. Prefetch to an enabled EBU location

If the EBU is enabled a prefetch may be transferred outside of the chip. Depending on the devices connected the prefetch may cause side effects (e.g. access to a modify-on-read register). No generic mitigation is possible and the side effects, if any, will depend on the specific implementation.

If the prefetch is to an EBU location outside of the mapped external device range then the following side effects will occur:

- An error interrupt will be signalled by the SRI Xbar bus system and the Xbar error register updated.

To mitigate against this either the Xbar unimplemented address interrupt generation can be disabled or the overall Xbar interrupt can be disabled

- XBAR_ARBCON6.PRERREN = 0_B (default after reset: 1_B)
- XBAR_SRC.SRE = 0_B

5. Prefetch to a disabled (TC1798) or unavailable (TC1791,TC1793) EBU

location

If the prefetch is to a disabled or unavailable EBU location then the following side effects will occur:

- An error interrupt will be signalled by the SRI Xbar bus system and the Xbar error register updated.

To mitigate against this either the Xbar unimplemented address interrupt generation can be disabled or the overall Xbar interrupt can be disabled

- XBAR_ARBCON6.PRERREN = 0_B (default after reset: 1_B)
- XBAR_SRC.SRE = 0_B

EBU_TC.023 Read Data Corruption when accessing SDRAM/DDRAM

When accessing SDRAM/DDR using the settings BUSRAPx.EXTCLOCK=11_B (or 10_B) and SDRMOD.BURSTL=011_B (100_B for DDR), reads can be prematurely terminated by an incorrectly scheduled REFRESH operation causing data corruption.

The combination of settings creates a `gap` in an internal busy signal between the read command and the read data returning. This gap allows a REFRESH command to commence before the read data is completely read. If BURSTL=011_B (burst length 8), the read does not have time to complete at the SDRAM device before the PRECHARGE command associated with the REFRESH operation terminates the access.

For burst lengths less than 8 (16 for DDR), the read will complete before it can be terminated by the PRECHARGE and the bug will not be triggered.

For clock ratios other than 1:4, the gap in the busy signal does not occur and the bug will not be triggered.

Workarounds

- Use shorter burst lengths (<16 for DDR or <8 for SDRAM), or
- Predivide the EBU internal clock (bit field EBU_CLC.EBUDIV) to avoid using an EBU Clock Divide Ratio of 1:4.

FADC_TC.005 Equidistant multiple channel-timers

The description is an example for timer_1 and timer_2, but can also affect all other combinations of timers.

Timer_1 and Timer_2 are running with different reload-values. Both timers should start conversions with the requirement of equidistant timing.

Problem description:

Timer_1 becomes zero and starts a conversion. Timer_2 becomes zero during this conversion is running and sets the conversion-request-bit of channel_2. At the end of the conversion for channel_1 this request initiates a start for channel_2. But the Timer_2 is reloaded only when setting the request-bit for channel_2 and is decremented during the conversion of channel_1.

The correct behavior would be a reload when the requested conversion (of channel_2) is started.

Therefore the start of conversion for channel_2 is delayed by maximum one conversion-time. After this delay it will be continued with equidistant conversion-starts. Please refer to the following figure.

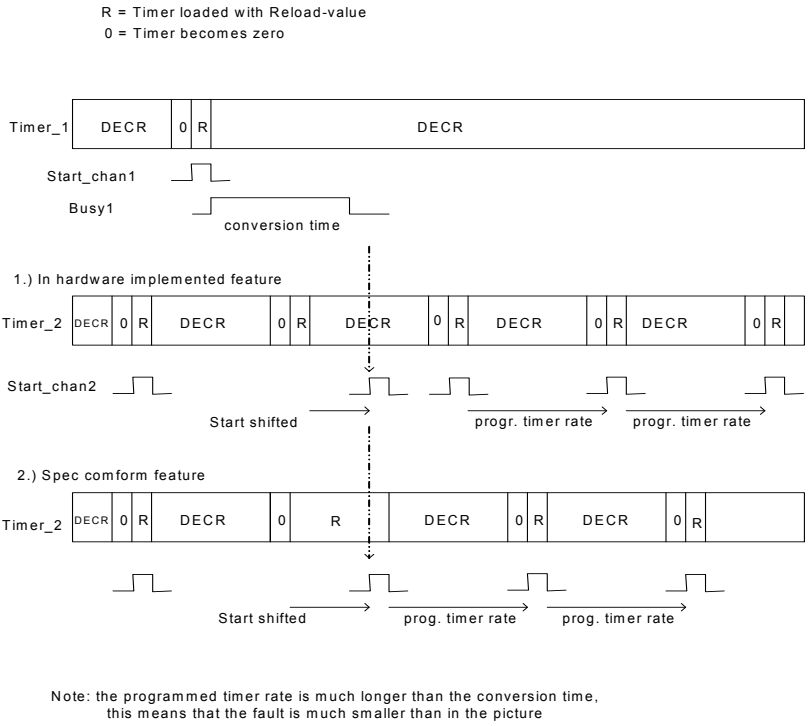

Figure 1 Timing concerning equidistant multiple timers

Workaround

Use one timer base in combination with neighboring trigger and selection by software which result has to be taken into account.

FlexRay_AI.087 After reception of a valid sync frame followed by a valid non-sync frame in the same static slot the received sync frame may be ignored

Description:

If in a static slot of an even cycle a valid sync frame followed by a valid non-sync frame is received, and the frame valid detection (prt_frame_decoded_on_X) of

the DEC process occurs one sclk after valid frame detection of FSP process (fsp_val_syncfr_chx), the sync frame is not taken into account by the CSP process (devte_xxs_reg).

Scope:

The erratum is limited to the case where more than one valid frame is received in a static slot of an even cycle.

Effects:

In the described case the sync frame is not considered by the CSP process. This may lead to a SyncCalcResult of MISSIMG_TERM (error flag SFS.MRCS set). As a result the POC state may switch to NORMAL_PASSIVE or HALT or the Startup procedure is aborted.

Workaround

Avoid static slot configurations long enough to receive two valid frames.

FlexRay_AI.088 A sequence of received WUS may generate redundant SIR.WUPA/B events

Description:

If a sequence of wakeup symbols (WUS) is received, all separated by appropriate idle phases, a valid wakeup pattern (WUP) should be detected after every second WUS.The E-Ray detects a valid wakeup pattern after the second WUS and then after each following WUS.

Scope:

The erratum is limited to the case where the application program frequently resets the appropriate SIR.WUPA/B bits.

Effects:

In the described case there are more SIR.WUPA/B events seen than expected.

Workaround

Ignore redundant SIR.WUPA/B events.

FlexRay_AI.089 Rate correction set to zero in case of SyncCalcResult=MISSING_TERM

Description:

In case a node receives too few sync frames for rate correction calculation and signals a SyncCalcResult of MISSING_TERM, the rate correction value is set to zero instead to the last calculated value.

Scope:

The erratum is limited to the case of receiving too few sync frames for rate correction calculation (SyncCalcResult=MISSING_TERM in an odd cycle).

Effects:

In the described case a rate correction value of zero is applied in NORMAL_ACTIVE / NORMAL_PASSIVE state instead of the last rate correction value calculated in NORMAL_ACTIVE state. This may lead to a desynchronisation of the node although it may stay in NORMAL_ACTIVE state (depending on gMaxWithoutClockCorrectionPassive) and decreases the probability to re-enter NORMAL_ACTIVE state if it has switched to NORMAL_PASSIVE (pAllowHaltDueToclock=false).

Workaround

It is recommended to set gMaxWithoutClockCorrectionPassive to 1. If missing sync frames cause the node to enter NORMAL_PASSIVE state, use higher level application software to leave this state and to initiate a re-integration into the cluster. HALT state can also be used instead of NORMAL_PASSIVE state by setting pAllowHaltDueToClock to true.

FlexRay_AI.090 Flag SFS.MRCS is set erroneously although at least one valid sync frame pair is received

Description:

If in an odd cycle 2c+1 after reception of a sync frame in slot n the total number of different sync frames per double cycle has exceeded gSyncNodeMax and the node receives in slot n+1 a sync frame that matches with a sync frame received in the even cycle 2c, the sync frame pair is not taken into account by CSP process. This may cause the flags SFS.MRCS and EIR.CCF to be set erroneously.

Scope:

The erratum is limited to the case of a faulty cluster configuration where different sets of sync frames are transmitted in even and odd cycles and the total number of different sync frames is greater than gSyncNodeMax.

Effects:

In the described case the error interrupt flag EIR.CCF is set and the node may enter either the POC state NORMAL_PASSIVE or HALT.

Workaround

Correct configuration of gSyncNodeMax.

FlexRay_AI.091 Incorrect rate and/or offset correction value if second Secondary Time Reference Point (STRP) coincides with the action point after detection of a valid frame

Description:

If a valid sync frame is received before the action point and additionally noise or a second frame leads to a STRP coinciding with the action point, an incorrect deviation value of zero is used for further calculations of rate and/or offset correction values.

Scope:

The erratum is limited to configurations with an action point offset greater than static frame length.

Effects:

In the described case a deviation value of zero is used for further calculations of rate and/or offset correction values. This may lead to an incorrect rate and/or offset correction of the node.

Workaround

Configure action point offset smaller than static frame length.

FlexRay_AI.092 Initial rate correction value of an integrating node is zero if pMicroInitialOffsetA,B = 0x00

Description:

The initial rate correction value as calculated in figure 8-8 of protocol spec v2.1 is zero if parameter pMicroInitialOffsetA,B was configured to be zero.

Scope:

The erratum is limited to the case where pMicroInitialOffsetA,B is configured to zero.

Effects:

Starting with an initial rate correction value of zero leads to an adjustment of the rate correction earliest 3 cycles later (see figure 7-10 of protocol spec v2.1). In a worst case scenario, if the whole cluster is drifting away too fast, the integrating node would not be able to follow and therefore abort integration.

Workaround

Avoid configurations with pMicroInitialOffsetA,B equal to zero. If the related configuration constraint of the protocol specification results in

pMicroInitialOffsetA,B equal to zero, configure it to one instead. This will lead to a correct initial rate correction value, it will delay the startup of the node by only one microtick.

FlexRay_AI.093 Acceptance of startup frames received after reception of more than gSyncNodeMax sync frames

Description:

If a node receives in an even cycle a startup frame after it has received more than gSyncNodeMax sync frames, this startup frame is added erroneously by process CSP to the number of valid startup frames (zStartupNodes). The faulty number of startup frames is delivered to the process POC. As a consequence this node may integrate erroneously to the running cluster because it assumes that it has received the required number of startup frames.

Scope:

The erratum is limited to the case of more than gSyncNodeMax sync frames.

Effects:

In the described case a node may erroneously integrate successfully into a running cluster.

Workaround

Use frame schedules where all startup frames are placed in the first static slots. gSyncNodeMax should be configured to be greater than or equal to the number of sync frames in the cluster.

FlexRay_AI.094 Sync frame overflow flag `EIR.SFO` may be set if slot counter is greater than 1024

Description:

If in the static segment the number of transmitted and received sync frames reaches gSyncNodeMax and the slot counter in the dynamic segment reaches

the value cStaticSlotIDMax + gSyncNodeMax = 1023 + gSyncNodeMax, the sync frame overflow flag `EIR.SFO` is set erroneously.

Scope:

The erratum is limited to configurations where the number of transmitted and received sync frames equals to gSyncNodeMax and the number of static slots plus the number of dynamic slots is greater or equal than 1023 + gSyncNodeMax.

Effects:

In the described case the sync frame overflow flag `EIR.SFO` is set erroneously. This has no effect to the POC state.

Workaround

Configure gSyncNodeMax to number of transmitted and received sync frames plus one or avoid configurations where the total of static and dynamic slots is greater than cStaticSlotIDMax.

FlexRay_AI.095 Register RCV displays wrong value

Description:

If the calculated rate correction value is in the range of [-pClusterDriftDamping .. +pClusterDriftDamping], vRateCorrection of the CSP process is set to zero. In this case register `RCV` should be updated with this value. Erroneously `RCV.RCV[11:0]` holds the calculated value in the range [-pClusterDriftDamping .. +pClusterDriftDamping] instead of zero.

Scope:

The erratum is limited to the case where the calculated rate correction value is in the range of [-pClusterDriftDamping .. +pClusterDriftDamping].

Effects:

The displayed rate correction value `RCV.RCV[11:0]` is in the range of [-pClusterDriftDamping .. +pClusterDriftDamping] instead of zero. The error of the displayed value is limited to the range of [-pClusterDriftDamping .. +pClusterDriftDamping]. For rate correction in the next double cycle always the correct value of zero is used.

Workaround

A value of `RCV.RCV[11:0]` in the range of [-pClusterDriftDamping .. +pClusterDriftDamping] has to be interpreted as zero.

FlexRay_AI.096 Noise following a dynamic frame that delays idle detection may fail to stop slot

Description:

If (in case of noise) the time between ’potential idle start on X’ and ’CHIRP on X’ (see Protocol Spec. v2.1, Figure 5-21) is greater than gdDynamicSlotIdlePhase, the E-Ray will not remain for the remainder of the current dynamic segment in the state ’wait for the end of dynamic slot rx’. Instead, the E-Ray continues slot counting. This may enable the node to further transmissions in the current dynamic segment.

Scope:

The erratum is limited to noise that is seen only locally and that is detected in the time window between the end of a dynamic frame’s DTS and idle detection (’CHIRP on X’).

Effects:

In the described case the faulty node may not stop slot counting and may continue to transmit dynamic frames. This may lead to a frame collision in the current dynamic segment.

Workaround

None.

FlexRay_AI.097 Loop back mode operates only at 10 MBit/s

Description:

The looped back data is falsified at the two lower baud rates of 5 and 2.5 MBit/s.

Scope:

The erratum is limited to test cases where loop back is used with the baud rate prescaler (`PRTC1.BRP[1:0]`) configured to 5 or 2.5 MBit/s.

Effects:

The loop back self test is only possible at the highest baud rate.

Workaround

Run loop back tests with 10 MBit/s (`PRTC1.BRP[1:0]` = 00_B).

FlexRay_AI.099 Erroneous cycle offset during startup after abort of start-up or normal operation

Description:

An abort of startup or normal operation by a READY command near the macotick border may lead to the effect that the state INITIALIZE_SCHEDULE is one macrotick too short during the first following integration attempt. This leads to an early cycle start in state INTEGRATION_COLDSTART_CHECK or INTEGRATION_CONSISTENCY_CHECK.

As a result the integrating node calculates a cycle offset of one macrotick at the end of the first even/odd cycle pair in the states INTEGRATION_COLDSTART_CHECK or INTEGRATION_CONSISTENCY_CHECK and tries to correct this offset.

If the node is able to correct the offset of one macrotick (pOffsetCorrectionOut >> gdMacrotick), the node enters NORMAL_ACTIVE with the first startup attempt.

If the node is not able to correct the offset error because pOffsetCorrectionOut is too small (pOffsetCorrectionOut ≤ gdMacrotick), the node enters

ABORT_STARTUP and is ready to try startup again. The next (second) startup attempt is not effected by this erratum.

Scope:

The erratum is limited to applications where READY command is used to leave STARTUP, NORMAL_ACTIVE, or NORMAL_PASSIVE state.

Effects:

In the described case the integrating node tries to correct an erroneous cycle offset of one macrotick during startup.

Workaround

With a configuration of pOffsetCorrectionOut >> gdMacrotick • (1+cClockDeviationMax) the node will be able to correct the offset and therefore also be able to successfully integrate.

FlexRay_AI.100 First WUS following received valid WUP may be ignored

Description:

When the protocol engine is in state WAKEUP_LISTEN and receives a valid wakeup pattern (WUP), it transfers into state READY and updates the wakeup status vector `CCSV.WSV[2:0]` as well as the status interrupt flags `SIR.WST` and `SIR.WUPA/B`. If the received wakeup pattern continues, the protocol engine may ignore the first wakeup symbol (WUS) following the state transition and signals the next `SIR.WUPA/B` at the third instead of the second WUS.

Scope:

The erratum is limited to the reception of redundant wakeup patterns.

Effects:

Delayed setting of status interrupt flags `SIR.WUPA/B` for redundant wakeup patterns.

Workaround

None.

FlexRay_AI.101 READY command accepted in READY state

Description:

The E-Ray module does not ignore a READY command while in READY state.

Scope:

The erratum is limited to the READY state.

Effects:

Flag CCSV.CSI is set. Cold starting needs to be enabled by POC command ALLOW_COLDSTART (SUCC1.CMD = 1001_B).

Workaround

None.

FlexRay_AI.102 Slot Status vPOC!SlotMode is reset immediately when entering HALT state

Description:

When the protocol engine is in the states NORMAL_ACTIVE or NORMAL_PASSIVE, a HALT or FREEZE command issued by the Host resets vPOC!SlotMode immediately to SINGLE slot mode (CCSV.SLM[1:0] = 00_B). According to the FlexRay protocol specification, the slot mode should not be reset to SINGLE slot mode before the following state transition from HALT to DEFAULT_CONFIG state.

Scope:

The erratum is limited to the HALT state.

Effects:

The slot status vPOC!SlotMode is reset to SINGLE when entering HALT state.

Workaround

None.

FlexRay_AI.103 Received messages not stored in Message RAM when in Loop Back Mode

After a FREEZE or HALT command has been asserted in NORMAL_ACTIVE state, and if state LOOP_BACK is then entered by transition from HALT state via DEF_CONFIG and CONFIG, it may happen that acceptance filtering for received messages is not started, and therefore these messages are not stored in the respective receive buffer in the Message RAM.

Scope:

The erratum is limited to the case where Loop Back Mode is entered after NORMAL_ACTIVE state was left by FREEZE or HALT command.

Effects:

Received messages are not stored in Message RAM because acceptance filtering is not started.

Workaround

Leave HALT state by hardware reset.

GPT12_TC.001 T2IN/T3IN Port Connections

The following connections listed in TC1798 User's Manual V1.1 are incorrect. The correct connections are listed below.

GPT120

- T2INB is connected to P2.12 (instead of P3.14)
- T3INB is connected to P2.8 (instead of P3.8)

GPT121

- T2INA is connected to P2.12 (instead of P3.14)
- T3INA is connected to P2.8 (instead of P3.8)

The corresponding connections listed in the Data Sheet are correct.

MSC_TC.009 Missing Receive Data Interrupt

A problem with receive data interrupt generation on the upstream channel occurs in a specific corner case if all of the following three conditions are met at the same time:

1. Option `ICR.RDIE` = 10_B is selected as interrupt generation condition (i.e. interrupt only if received data is not equal to 00_H), and
2. Two MSC frames (F_n, F_{n+1}) are transmitted on the upstream channel in series (i.e. after two stop bits immediately a start bit occurs), and
3. The leading edge of the start bit generated by the transmitter arrives at input SDI of the microcontroller before the end or at the boundary of its MSC stop bit cycle. This is typically the case when the transmitter uses a clock that is independent of the microcontroller clock, and the actual baud rate of the transmitter is higher or equal to the configured baud rate of the microcontroller (within the permitted range for asynchronous transfers).

In this case, the interrupt request at the end of frame F_n will not be generated and flag `ISR.URDI` will not be set.

Workarounds

1. Use a frame with a third stop bit.
2. Do not use the interrupt generation condition `ICR.RDIE` = 10_B (depending on data bits of received frame). Use e.g. `ICR.RDIE` = 01_B, and test for received data equal to zero by software.

OCDS_TC.035 Debug reset will not disable the OCDS

Debug reset will not clear CBS_OSTATE.OEN so OCDS stays enabled. DBGSR.SUSP is not cleared as well.

Workaround

Disable the OCDS if needed after Debug reset by writing CBS_OEC.DS. Write reset value to DBGSR e.g. to start the CPU.

PCP_TC.023 JUMP sometimes takes an extra cycle

Following a taken JUMP, the main state machine may misleadingly take an additional cycle of pause. This occurs if the already prefetched next or second next instruction after the JUMP is one of the following instructions:

- LD.P
- ST.P
- DEBUG
- Any instruction with extension .PI

This does not cause any different program flow or incorrect result, it just adds an extra dead cycle.

Workaround

None.

PCP_TC.032 Incorrect PCP behaviour following FPI timeouts (as a slave)

When PRAM is being accessed from the FPI bus and an FPI time-out occurs then this can lead to corruption or loss of the current and subsequent FPI accesses. In general an FPI time-out during an access to the PCP is unlikely since FPI time-out is usually programmed for a large number of FPI clock cycles and the only time that the FPI access cannot be immediately responded to by the PCP is during the execution of atomic PRAM instructions. FPI accesses are locked out for the entire duration of any sequence of back to back atomic PRAM instructions. The combination of a low FPI time-out setting and long sequences of atomic PRAM instructions could therefore result in FPI time-out.

Workaround

Keep the FPI time-out setting as high as possible and do not include long sequences of back to back atomic PRAM instructions. If N is the highest amount of back to back atomic PRAM instructions in any PCP channel program, FPI time-out should at least be 10 times N.

PCP_TC.041 Coincident PCP non-atomic Write and FPI RMW Access to PRAM

A PCP non-atomic PRAM write access coinciding with an FPI read/modify/write (rmw) to PRAM issued by an external master will break the atomicity of the FPI rmw instruction.

PCP non-atomic writes include instructions ST.P, ST.PI and channel context save operations.

Note: All operations will work correctly, i.e. no lost stores etc., as long as the PRAM write addresses are different.

Workaround 1

Replace the non-atomic PRAM instructions with atomic ones (e.g. XCH.PI, or MCLR/MSET.PI for semaphore operations).

If backward compatibility with other devices of the AudoXY microcontroller families is an issue/requirement, then use the following workaround:

Workaround 2

In case the PCP needs to write to the same PRAM address as an external FPI rmw instruction (i.e. a semaphore operation), replace these non-atomic store PRAM instructions with FPI based store instructions such as ST.F, ST.FI.

Alternatively, variables shared by PCP and external masters may be located in RAM areas other than PRAM, such that software tools implicitly use the FPI based store instructions such as ST.F, ST.FI.

SRI_TC.001 Atomic CPU operation with Bus Address ECC Error will cause bus hang

During any atomic CPU operation (LDMST, SWAP.W, ST.T) targeting a module on the SPB bus, if a SRI bus address ECC error occurs during the write then the next access to the same slave will cause a bus hang.

The ECC error will be promptly notified to the SCU as a trap request. However, the CPU may not be able to service it if the bus is blocked.

Workaround

This is considered to be an exceptional rare error event (caused e.g. by alpha particles) and can be recovered by the watchdog timer.

SRI_TC.002 SRI_XBAR Debug registers may capture wrong transaction during pipelining

The XBAR_SRI module should capture information about the first transaction that was finished with an error e.g. due to an access to a reserved address. In some situations it may happen that information of a different transaction are captured.

Every new address phase is stored in the debug registers (`XBAR_ERRx`, `XBAR_ERRADDRx`) unless they have been locked by a previous debug event. In this case, the debug event is a protocol error that occurs during the data phase. If another transactions address phase is pipelined before the first transactions data phase occurs, the second transaction information will be locked into the debug registers instead of the transaction that caused the protocol error.

Note: This only affects debug functionality and is without effect on normal device operation.

Workaround

None.

SSC_AI.022 Phase error detection switched off too early at the end of a transmission

The phase error detection will be switched off too early at the end of a transmission. If the phase error occurs at the last bit to be transmitted, the phase error is lost.

Workaround

Don’t use the phase error detection.

SSC_AI.023 Clock phase control causes failing data transmission in slave mode

If SSC_CON.PH = 1 and no leading delay is issued by the master, the data output of the slave will be corrupted. The reason is that the chip select of the master enables the data output of the slave. As long as the chip is inactive the slave data output is also inactive.

Workaround

A leading delay should be used by the master.

A second possibility would be to initialize the first bit to be sent to the same value as the content of PISEL.STIP.

SSC_AI.024 SLSO output gets stuck if a reconfig from slave to master mode happens

The slave select output SLSO gets stuck if the SSC will be re-configured from slave to master mode. The SLSO will not be deactivated and therefore not correct for the 1st transmission in master mode. After this 1st transmission the chip select will be deactivated and working correctly for the following transmissions.

Workaround

Ignore the 1st data transmission of the SSC when changed from slave to master mode.

SSC_AI.025 First shift clock period will be one PLL clock too short because not syncronized to baudrate

The first shift clock signal duration of the master is one PLL clock cycle shorter than it should be after a new transmit request happens at the end of the previous transmission. In this case the previous transmission had a trailing delay and an inactive delay.

Workaround

Use at least one leading delay in order to avoid this problem.

SSC_AI.026 Master with highest baud rate set generates erroneous phase error

If the SSC is in master mode, the highest baud rate is initialized and CON.PO = 1 and CON.PH = 0 there will be a phase error on the MRST line already on the shift edge and not on the latching edge of the shift clock.

- Phase error already at shift edge
  The master runs with baud rate zero. The internal clock is derived from the rising and the falling edge. If the baud rate is different from zero there is a gap between these pulses of these internal generated clocks.
  However, if the baud rate is zero there is no gap which causes that the edge detection is to slow for the "fast" changing input signal. This means that the input data is already in the first delay stage of the phase detection when the delayed shift clock reaches the condition for a phase error check. Therefore the phase error signal appears.
- Phase error pulse at the end of transmission
  The reason for this is the combination of point 1 and the fact that the end of the transmission is reached. Thus the bit counter SSCBC reaches zero and the phase error detection will be switched off.

Workaround

Don’t use a phase error in master mode if the baud rate register is programmed to zero (SSCBR = 0) which means that only the fractional divider is used.
Or program the baud rate register to a value different from zero (SSCBR > 0) when the phase error should be used in master mode.

3 Deviations from Electrical- and Timing Specification

AC_PER_TC.P001 Driver Settings for SSC and MLI Outputs on A2 Pads

In general, AC timings are specified for the strongest/fastest possible output driver setting.

However, for SSC and MLI, the specified timings are only valid if outputs on A2 pads are configured for strong driver/medium edge mode.

PLL_ERAY_TC.P001 PLL lock-in time

The minimum value for parameter t_L (PLL lock-in time) for ERAY Phase Locked Loop (PLL_ERAY) is 5.6µs instead of 50µs.

PLL_TC.P006 PLL lock-in time

The minimum value for parameter t_L (PLL lock-in time) for Phase Locked Loop (PLL) is 14µs

- instead of 50µs for N > 32
- instead of 100µs for N ≤ 32.

4 Application Hints

ADC_AI.H002 Minimizing Power Consumption of an ADC Module

For a given number of A/D conversions during a defined period of time, the total energy (power over time) required by the ADC analog part during these conversions via supply V_{DDM} is approximately proportional to the converter active time.

Recommendation for Minimum Power Consumption:

In order to minimize the contribution of A/D conversions to the total power consumption, it is recommended

1. to select the internal operating frequency of the analog part (f_{ADCI} or f_{ANA}, respectively)[1] near the **maximum** value specified in the Data Sheet, and
2. to switch the ADC to a power saving state (via ANON) while no conversions are performed. Note that a certain wake-up time is required before the next set of conversions when the power saving state is left.

Note: The selected internal operating frequency of the analog part that determines the conversion time will also influence the sample time t_S. The sample time t_S can individually be adapted for the analog input channels via bit field STC.

ADC_AI.H003 Injected conversion may be performed with sample time of aborted conversion

For specific timing conditions and configuration parameters, a higher prioritized conversion c_i (including a synchronized request from another ADC kernel) in cancel-inject-repeat mode may erroneously be performed with the sample time parameters of the lower prioritized cancelled conversion c_c. This can lead to

1) Symbol used depends on product family: e.g. f_{ANA} is used in the documentation of devices of the AUDO-NextGeneration family.

wrong sample results (depending on the source impedance), and may also shift the starting point of following conversions.

The conditions for this behavior are as follows (all 3 conditions must be met):

1. **Sample Time setting**: injected conversion c_i and cancelled conversion c_c use different sample time settings, i.e. bit fields `STC` in the corresponding Input Class Registers `INPCRx` (for c_c) and `INPCRy` (for c_i) are programmed to different values.
2. **Timing condition**: conversion c_i starts during the first f_{ADCI} clock cycle of the sample phase of c_c.
3. **Configuration parameters**: the ratio between the analog clock f_{ADCI} and the arbiter speed is as follows:

   $$N_A > N_D*(N_{AR}+3),$$

   with

   a) N_A = ratio f_{ADC}/f_{ADCI} (N_A = 4 .. 63, as defined in bit field `DIVA`),

   b) N_D = ratio f_{ADC}/f_{ADCD} = number of f_{ADC} clock cycles per arbitration slot (N_D = 1 .. 4, as defined in bit field `DIVD`),

   c) N_{AR} = number of arbitration slots per arbitration round (N_{AR} = 4, 8, 16, or 20, as defined in bit field `ARBRND`).

All bit fields mentioned above are located in register `GLOBCTR`.

As can be seen from the formula above, a problem typically only occurs when the arbiter is running at maximum speed, and a divider $N_A > 7$ is selected to obtain f_{ADCI}.

Recommendation 1

Select the same sample time for injected conversions c_i and potentially cancelled conversions c_c, i.e. program all bit fields `STC` in the corresponding Input Class Registers `INPCRx` (for c_c) and `INPCRy` (for c_i) to the same value.

Recommendation 2

Select the parameters in register `GLOBCTR` according to the following relation:

$$N_A \leq N_D*(N_{AR}+3).$$

BROM_TC.H002 Enabling CAN Communication in CAN Bootstrap Loader Mode

After completion of the download in CAN bootstrap loader mode, the module clock f_{CLC} is disabled. Therefore, code executed after download in CAN bootstrap loader mode can not directly continue communication via the CAN interface. It first needs to initialize register CAN_CLC to enable f_{CLC}.

CPU_TC.H005 Wake-up from Idle/Sleep Mode

A typical use case for idle or sleep mode is that software puts the CPU into one of these modes each time it has to wait for an interrupt.

Idle or Sleep Mode is requested by writing to the Power Management Control and Status Register (PMCSR). However, when the write access to PMCSR is delayed e.g. by a higher priority bus access, TriCore may enter idle or sleep mode while the interrupt which should wake up the CPU is already executed. As long as no additional interrupts are triggered, the CPU will endlessly stay in idle/sleep mode.

Therefore, e.g. the following software sequence is recommended (for user mode 1, supervisor mode):

```
_disable();           // disable interrupts
do {
SCU_PMCSR = 0x1;      // request idle mode
if( SCU_PMCSR );      // ensure PMCSR is written

_enable();            // after wake-up: enable interrupts
_disable();           // after service: disable interrupts
} while( !condition ); // return to idle mode depending on
                      // condition set by interrupt handler
_enable();
```

CPU_TC.H006 Store Buffering in TC1.6/P/E Processors

Overview

Store buffering is a method of increasing processor performance by decoupling memory write operations from the instruction execution flow within the CPU. All write data is placed in a FIFO buffer (known as the store buffer) by the CPU prior to being read by the memory/bus interfaces and written to memory. This allows the processor to continue execution without waiting for the write data to be written to the target memory location. Data is written to the store buffer at processor speed and read from the store buffer at memory/bus speed. Typically the read bandwidth from the store buffer will exceed the write bandwidth from the processor, only if the store buffer fills will the processor stall.

To further increase performance memory read operations are prioritised ahead of memory write operations from the store buffer. This ensures that the processor does not stall on data loads while data writes are pending in the store buffer. A side effect of this prioritising is that memory may not be accessed in program order.

Operational Details

The function of the store buffer is designed to be invisible to the end user under normal operation:

- All CPU load operations are checked against the store buffer contents. Data for matching load addresses is either immediately forwarded to the CPU from the store buffer (TC1.6, TC1.6P) or written to memory prior to the load operation proceeding (TC1.6E).
- All loads and store operations to peripheral regions (typically segments E_H and F_H) are performed in strict program order (no load prioritisation).

The operation of the store buffer can become visible when in-order memory access is required to non-peripheral segments.

This can occur under the following circumstances:

- When programming flash memory.
- When performing memory testing with the processor.
- When data is required to be in memory for inter-core/inter-module communication.

In such cases the following solutions may be employed:

- The store buffer may be explicitly flushed by use of a DSYNC instruction.
- The store buffer may be disabled by setting `SMACON.IODT`. This should not be done during normal operation as it significantly impacts performance.

Examples

The following examples refer to memory accesses to non-peripheral regions (i.e. segments 0_H .. D_H):

Example-1a Out of order memory access due to load prioritisation

```
Program Flow     -     Memory Access
 st-1                   ld-4
 st-2                   ld-5
 st-3                   ld-6
 ld-4                   st-1
 ld-5                   st-2
 ld-6                   st-3
```

Example-1b In order memory access enforced by DSYNC

```
Program Flow     -     Memory Access
 st-1                   st-1
 st-2                   st-2
 st-3                   st-3
 dsync
 ld-4                   ld-4
 ld-5                   ld-5
 ld-6                   ld-6
```

Example-2a Load forwarding from store buffer - no memory read (TC1.6/1.6P)

```
Program Flow     -     Memory Access
 st.w [a0], d0
 ld.w d1,   [a0]        st.w [a0], d0
```

Example-2b In order memory access enforced by DSYNC (TC1.6/1.6P)

```
Program Flow     -     Memory Access
 st.w [a0], d0          st.w [a0], d0
 dsync
 ld.w d1,   [a0]        ld.w d1,   [a0]
```

EBU_TC.H010 EBU SFR access for Non-TriCore masters after reset

The intention of bit EBU_MODCON.ACCSINH is to prevent masters other than the Tricore from accessing an unconfigured EBU. The reset value of bit EBU_MODCON.ACCSINH is 1_B.

However, bit ACCSINH = 1_B also prevents other masters such as Cerberus from accessing the SFR range, not only from external memory regions.

Workaround

Code running on TriCore has to set bit EBU_MODCON.ACCSINH = 0_B to disable the inhibit feature before any other master attempts to access external memory or the SFR range.

EBU_TC.H011 Hints for DDR PCB Layout

The primary use case intended for the DDR interface is to allow the EBU output clock frequency to be lowered to reduce EMI without impacting the achievable data rate.

The DDR interface uses the common EBU pins. However, if DDR memory is to be used, it should not be used simultaneously with different memory types as differential loads on the data bus byte lanes and the associated DQS pins must be avoided if the interface is to operate correctly.

The DDR interface is intended to be used with source series termination of PCB traces using resistors of between 18 and 33 ohms. For this to operate correctly, PCB traces must be kept as short as possible (20mm should be considered as a target for maximum length) and PCB traces must be designed to eliminate as many reflections as possible. This is particularly critical for the clock signals (including DQS) where reflections arriving at the input when the signal is

passing through the threshold region could cause unintended clock edges and capture incorrect data. As the address and control signals are only single data rate, source series termination of the PCB traces may not be required for these signals.

The high frequency switching of the DDR interface will stress the pad power supply. Please pay particular attention to power and ground supply connections to the EBU pads to avoid excessive ground bounce and/or dips in the VDDEBU supply causing transient, invalid logic levels on the EBU outputs.

Further recommendations:

- All traces should have the same length. The length matching helps to reduce different delay times of signal paths.
- All traces should have the same ground reference plane. It is not recommended to change the ground reference layer of data line groups.
- The ground reference plane should not be interrupted by any vias or other cuts. The return current path should have a solid ground.

FlexRay_AI.H004 Only the first message can be received in External Loop Back mode

If the loop back (TXD to RXD) will be performed via external physical transceiver, there will be a large delay between TXD and RXD.

A delay of two sample clock periods can be tolerated from TXD to RXD due to a majority voting filter operation on the sampled RXD.

Only the first message can be received, due to this delay.

To avoid that only the first message can be received, a start condition of another message (idle and sampling '0' -> low pulse) must be performed.

The following procedure can be applied at one or both channels:

- wait for no activity (`TEST1.AOx`=0 -> bus idle)
- set Test Multiplexer Control to I/O Test Mode (`TEST1.TMC`=2), simultaneously `TXDx`=`TXENx`=0
- wait for activity (`TEST1.AOx`=1 -> bus not idle)
- set Test Multiplexer Control back to Normal signal path (`TEST1.TMC`=0)
- wait for no activity (`TEST1.AOx=0` -> bus idle)

Now the next transmission can be requested.

FlexRay_AI.H005 Initialization of internal RAMs requires one eray_bclk cycle more

The initialization of the E-Ray internal RAMs as started after hardware reset or by CHI command CLEAR_RAMS (SUCC1.CMD[3:0] = 1100_B) takes 2049 eray_bclk cycles instead of 2048 eray_bclk cycles as described in the E-Ray Specification.

Signalling of the end of the RAM initialization sequence by transition of MHDS.CRAM from 1_B to 0_B is correct.

FlexRay_AI.H006 Transmission in ATM/Loopback mode

When operating the E-Ray in ATM/Loopback mode there should be only one transmission active at the same time. Requesting two or more transmissions in parallel is not allowed.

To avoid problems, a new transmission request should only be issued when the previously requested transmission has finished. This can be done by checking registers TXRQ1/2/3/4 for pending transmission requests.

FlexRay_AI.H007 Reporting of coding errors via TEST1.CERA/B

When the protocol engine receives a frame that contains a frame CRC error as well as an FES decoding error, it will report the FES decoding error instead of the CRC error, which should have precedence according to the non-clocked SDL description.

This behaviour does not violate the FlexRay protocol conformance. It has to be considered only when TEST1.CERA/B is evaluated by a bus analysis tool.

FlexRay_AI.H009 Return from test mode operation

The E-Ray FlexRay IP-module offers several test mode options

- Asynchronous Transmit Mode
- Loop Back Mode
- RAM Test Mode
- I/O Test Mode

To return from test mode operation to regular FlexRay operation we strongly recommend to apply a hardware reset via input eray_reset to reset all E-Ray internal state machines to their initial state.

Note: The E-Ray test modes are mainly intended to support device testing or FlexRay bus analyzing. Switching between test modes and regular operation is not recommended.

FlexRay_AI.H010 Driver SW must launch CLEAR_RAMS command before reading from E-Ray RAMs

After a Power-on-Reset, the RAMs used by the E-Ray module must be written once. The recommended solution is to trigger a “CLEAR_RAMS” command (via register SUCC1). “CLEAR_RAMS” fills a defined value into all memory locations.

An alternate solution is to write explicitly by SW to all RAM locations, which are intended to be read later, e.g. by writing the complete configuration and writing into all allocated message buffers, including receive buffers. The latter activity may be required if buffers are configured to store frames sent in the dynamic segment. The sent frames may be smaller than the configured buffer size. If the SW reads the amount of configured data (not the amount of received data), it may read from non-activated RAM locations.

Reading from RAM locations before at least writing once to it may cause a Parity Error Trap (TC1797) or an ECC Error Trap.

GPT12_AI.H001 Modification of Block Prescalers BPS1 and BPS2

The block prescalers BPS1 and BPS2, controlled via bit fields T3CON.BSP1 and T6CON.BPS2, determine the basic clock for the GPT1 and GPT2 block, respectively.

After reset, when initializing a block prescaler BPSx to a value different from its default value (00_B), it must be initialized first before any mode involving external trigger signals is configured for the associated GPTx block. These modes include counter, incremental interface, capture, and reload mode. Otherwise, unintended count/capture/reload events may occur.

In case a block prescaler BPSx needs to be modified during operation of the GPTx block, disable related interrupts before modification of BPSx, and afterwards clear the corresponding service request flags and re-initialize those registers (T2, T3, T4 in block GPT1, and T5, T6, CAPREL in block GPT2) that might be affected by an unintended count/capture/reload event.

GPTA_TC.H004 Handling of GPTA Service Requests

Concerning the relations between two events (request_1, request_2) from different service request sources that belong to the same service request group y of the GPTA module, two standard cases (1, 2) and one corner case can be differentiated:

Case 1

When request_2 is generated **before** the previous request_1 has been acknowledged, the common Service Request Flag SRR of service request group y is cleared after request_1 is acknowledged. Since the occurrence of request_1 and request_2 is also flagged in the Service Request State Registers SRS*,[1)] all request sources can be identified by reading SRS* in the interrupt service routine or PCP channel program, respectively.

Case 2

When request_2 is generated **after** request_1 has been acknowledged, both flag SRR and the associated flag for request_2 in register SRS* are set, and the interrupt service routine/PCP channel program will be invoked again.

1) SRS* = abbreviation for Service Request State Registers SRSCn or SRSSn.

Corner Case

When request_2 is generated while request_1 is in the **acknowledge phase**, and the service routine/PCP channel program triggered by request_1 is reading register SRS* to determine the request source, then the following scenario may occur:

Depending on the relations between module clock f_{GPTA}, FPI-Bus clock, and the number of cycles required until the instruction reading SRS* is executed, the value read from SRS* may not yet indicate request_2, but only request_1 (unlike case 1). On the other hand, flag SRR (cleared when request_1 was acknowledged) is not set to trigger service for request_2 (unlike case 2).

As a consequence, recognition and service of request_2 will be delayed until the next request of one of the sources connected to this service request group y is generated.

Identification of Affected Systems

A system will **not** be affected by the corner case described above when the following condition is true:

(1a) READ - ACK ≥ max(icu, (N-1)*FPIDIV) for FDR in Normal Mode, or

(1b) READ - ACK ≥ max(icu, N*FPIDIV) for FDR in Fractional Mode

with:

- READ = number of f_{CPU}[1)] or f_{PCP} cycles between interrupt request (at CPU/PCP site) and register SRS* read operation.
  Number of cycles depends on implementation of service routine. “Worst case” with respect to corner case is minimum time:
  - READ = R_0 = 10 if instruction reading SRS* is directly located at entry point in Interrupt Vector Table in CPU Interrupt Service (sub-)routine
  - READ = R_1 = 14 if instruction reading SRS* is first instruction in CPU Interrupt Service (sub-)routine
  - Read = R_P = 16 if instruction reading SRS* is first instruction in PCP channel program

1) $f_{CPU} = f_{LMB}$ or f_{SRI}, depending on bus structure used in specific product.

  - R_X: number of extra f_{CPU} or f_{PCP} cycles to be added to R_0, R_1, or R_P, respectively, in case instruction reading SRS* is not the first instruction in the corresponding service routine.
- ACK = number of f_{CPU} or f_{PCP} cycles between interrupt request (at CPU/PCP site) and clearing of request flag SRR
  - ACK = 7 = constant for TriCore and PCP under all conditions (independent from ICU/PICU configuration)
- icu = clock ratio between ICU and CPU clock
  - icu = 2 with bit ICR.CONECYC=1_B, icu = 4 with bit ICR.CONECYC=0_B
- N = “maximum integer value” of clock ratio f_{FPI} / f_{GPTA}
  - N = 1024 - STEP for Normal Divider mode (DM = 01_B)
  - N = (1024 DIV STEP) + 1 for Fractional Divider mode (DM = 10_B), where DIV means “integer division”
- FPIDIV = clock ratio f_{CPU} / f_{FPI} for CPU and f_{PCP} / f_{FPI} for PCP

Example 1

PCP reads register SRS* with first instruction, GPTA is configured with fractional divider, STEP = $E4_H$, CONECYC = 0_B, FPIDIV = 2 (f_{PCP} = 2*f_{FPI})

This results in:

16 - 7 ≥ max(4, (1024 DIV 228 + 1)*2), or

9 ≥ max(4, (5*2)), or

9 ≥ max(4, 10), where max(4, 10) = 10

i.e. 9 ≥ 10 is false

i.e. this configuration is critical with respect to the corner case described above.

Example 2

PCP reads register SRS* with first instruction, GPTA is configured with fractional divider, STEP = $38E_H$, CONECYC = 0_B, FPIDIV = 2 (f_{PCP} = 2*f_{FPI})

This results in:

16 - 7 ≥ max(4, (1024 DIV 910 + 1)*2), or

9 ≥ max(4, (2*2)), or

9 ≥ max(4, 4), where max(4, 4) = 4

i.e. 9 ≥ 4 is true

i.e. this configuration is not affected by the corner case described above.

Recommendation

In case a system is affected by the corner case described above, the service routine/PCP channel program should read the status flags in SRS* again ≥ 1 GPTA module clock cycle after the first read operation to ensure earliest possible recognition of all events, e.g.:

```
Service Routine/PCP Program Entry:
 - Read SRS*
  - if flag is set: handle requesting source, clear
      corresponding flag via register SRSCx
  - Ensure elapsed time to next read of SRS* in Loop is
      ≥ 1 GPTA module clock cycle since routine entry
  Loop:
    - Read SRS*, exit if all flags are 0
    - Handle requesting source(s), clear
      corresponding flag(s) via register SRSCx
```

or (when the GPTA module clock is relatively high) e.g.:

```
Service Routine/PCP Program Entry:
 - Ensure time to first read of SRS* in Loop is
      ≥ 1 GPTA module clock cycle since routine entry
  Loop:
    - Read SRS*, exit if all flags are 0
    - Handle requesting source(s), clear
      corresponding flag(s) via register SRSCx
```

Note: In case the condition in formula (1a) or (1b) is not true, it would be possible to add n ≥ Rx + FPIDIV - 1 NOPs (+ ISYNC for CPU) at the beginning of the service routine to extend the time until SRS is read.*
Referring to Example 1 (Rx ≥ 1 cycle is missing, FPIDIV = 2), n ≥ 2 NOPs may be added before SRS is read to make this configuration uncritical.*
Make sure the NOPs are not eliminated by code optimizations.
However, basically it is still recommended to follow the general hint in paragraph “Recommendation” to improve code portability and become independent of cycle counting for individual configurations.

INT_TC.H003 Safe Cancellation of Service Requests

In specific cases, there might be the need to cancel a Service Request from a node SRNx before it is serviced by CPU or PCP (e.g. when two interrupt sources share the same Service Request Node).

Depending on the synchronization between CPU/PCP and the arbitration performed by the Interrupt Control Unit (ICU), it may happen that a request from SRNx is still serviced after its request flag SRR has been cleared.

Application Note AP32009 “Safe Cancellation of Service Requests” discusses these aspects in detail, including practical implementation examples. It is available in category ‘Documents’ at **www.infineon.com/32-bit-microcontrollers** and **www.myInfineon.com**.

LMU_TC.H001 Behaviour of Error Flags in Register LMU_MEMCON

The error status flags ADDER and DBERR in register LMU_MEMCON indicate errors detected during accesses to the LMU SRAM, the OLDA region, and to EMEM (in Emulation Device).

When clearing these flags by software (e.g. in an error handler routine), there is a remote possibility for a race condition if another SRI address phase ECC error or LMU memory related double bit error occurs before the respective flag is effectively cleared. In this case, the error handling routine would be invoked again as expected, but find that the corresponding ADDER or DBERR status flag is not set.

MSC_TC.H007 Start Condition for Upstream Channel

The reception of the upstream frame is started when a falling edge (1-to-0 transition) is detected on the SDI line.

In addition, reception is also started when a low level is detected on the SDI line while the upstream channel was in idle state, i.e.

- when the upstream channel is switched on (bit field URR in register USR is set to a value different from 000_B) and the SDI line is already on a low level, or

- after a frame has been received, and the SDI line is on a low level at the end of the last stop bit time slot (e.g. when the SDI line is permanently held low).

Therefore, make sure that the SDI line is pulled high (e.g. with an internal or external pull-up) while no transmission is performed.

MSC_TC.H010 Configuration of SCU.EMSR for the EMGSTOPMSC Signal

The emergency stop input signal EMGSTOPMSC of the MSC module is connected to the output signal of the emergency stop control logic located in the SCU. Its functionality is controlled by the SCU emergency stop register `SCU.EMSR`.

The emergency stop input line EMGSTOPMSC is used to indicate an emergency stop condition of a power device. In emergency case, shift register bits can be loaded bit-wise from the downstream data register instead from the ALTINL and ALTINH buses.

The emergency stop frame is sent out at each trigger event as long as the emergency stop signal is active. This means that in data repetition mode the emergency stop frame is repeatedly sent as long as the emergency stop signal is active.

Note: If the emergency stop signal is used by the MSC module with setting SCU.EMSR.MODE = 1_B (Asynchronous Mode), there is some low probability that the first emergency stop frame could be corrupted, but the following emergency stop frames will be correct.

Recommendation

- If the emergency stop signal is used by the MSC module, setting `SCU.EMSR.MODE` = 0_B (Synchronous Mode) is mandatory.
- Setting `SCU.EMSR.MODE` = 1_B (Asynchronous Mode) is not allowed to be used with the MSC module.

MultiCAN_AI.H005 TxD Pulse upon short disable request

If a CAN disable request is set and then canceled in a very short time (one bit time or less) then a dominant transmit pulse may be generated by MultiCAN module, even if the CAN bus is in the idle state.

Example for setup of the CAN disable request:

`CAN_CLC.DISR` = 1 and then `CAN_CLC.DISR` = 0

Workaround

Set all INIT bits to 1 before requesting module disable.

MultiCAN_AI.H006 Time stamp influenced by resynchronization

The time stamp measurement feature is not based on an absolute time measurement, but on actual CAN bit times which are subject to the CAN resynchronization during CAN bus operation.The time stamp value merely indicates the number of elapsed actual bit times. Those actual bit times can be shorter or longer than nominal bit time length due to the CAN resynchronization events.

Workaround

None.

MultiCAN_AI.H007 Alert Interrupt Behavior in case of Bus-Off

The MultiCAN module shows the following behavior in case of a bus-off status:

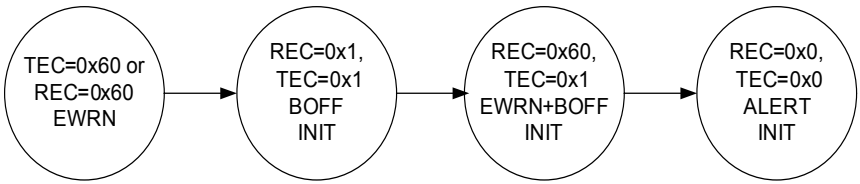

Figure 2 Alert Interrupt Behavior in case of Bus-Off

When the threshold for error warning (EWRN) is reached (default value of Error Warning Level EWRN = 0x60), then the EWRN interrupt is issued. The bus-off (BOFF) status is reached if TEC > 255 according to CAN specification, changing the MultiCAN module with REC and TEC to the same value 0x1, setting the INIT bit to 1_B, and issuing the BOFF interrupt. The bus-off recovery phase starts automatically. Every time an idle time is seen, REC is incremented. If REC = 0x60, a combined status EWRN+BOFF is reached. The corresponding interrupt can also be seen as a pre-warning interrupt, that the bus-off recovery phase will be finished soon. When the bus-off recovery phase has finished (128 times idle time have been seen on the bus), EWRN and BOFF are cleared, the ALERT interrupt bit is set and the INIT bit is still set.

MultiCAN_AI.H008 Effect of CANDIS on SUSACK

When a CAN node is disabled by setting bit NCR.CANDIS = 1_B, the node waits for the bus idle state and then sets bit NSR.SUSACK = 1_B.

However, SUSACK has no effect on applications, as its original intention is to have an indication that the suspend mode of the node is reached during debugging.

MultiCAN_TC.H002 Double Synchronization of receive input

The MultiCAN module has a double synchronization stage on the CAN receive inputs. This double synchronization delays the receive data by 2 module clock cycles. If the MultiCAN is operating at a low module clock frequency and high CAN baudrate, this delay may become significant and has to be taken into account when calculating the overall physical delay on the CAN bus (transceiver delay etc.).

MultiCAN_TC.H003 Message may be discarded before transmission in STT mode

If MOFCRn.STT=1 (Single Transmit Trial enabled), bit TXRQ is cleared (TXRQ=0) as soon as the message object has been selected for transmission and, in case of error, no retransmission takes places.

Therefore, if the error occurs between the selection for transmission and the real start of frame transmission, the message is actually never sent.

Workaround

In case the transmission shall be guaranteed, it is not suitable to use the STT mode. In this case, MOFCRn.STT shall be 0.

MultiCAN_TC.H004 Double remote request

Assume the following scenario: A first remote frame (dedicated to a message object) has been received. It performs a transmit setup (TXRQ is set) with clearing NEWDAT. MultiCAN starts to send the receiver message object (data frame), but loses arbitration against a second remote request received by the same message object as the first one (NEWDAT will be set).

When the appropriate message object (data frame) triggered by the first remote frame wins the arbitration, it will be sent out and NEWDAT is not reset. This leads to an additional data frame, that will be sent by this message object (clearing NEWDAT).

There will, however, not be more data frames than there are corresponding remote requests.

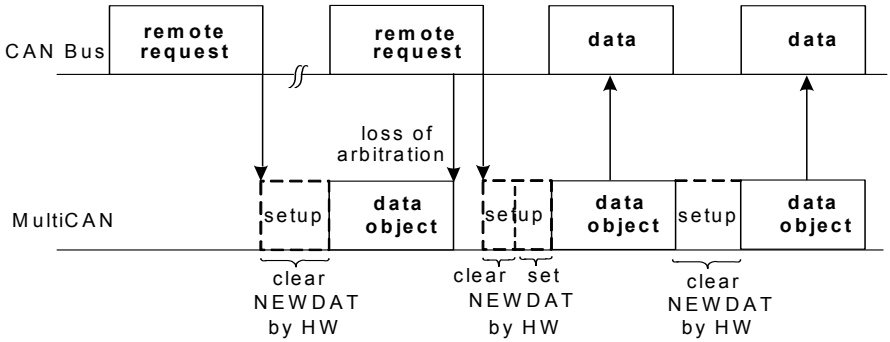

Figure 3 Loss of Arbitration

OCDS_TC.H001 IOADDR may increment after aborted IO_READ_BLOCK

If an IO_READ_BLOCK instruction is aborted by the host (switching the TAP controller to the update-DR state before enough data bits have been shifted out) it may happen under certain clock ratios that the IOADDR register is incremented nevertheless. This will result in an access to the wrong data in the succeeding IO_READ_* or IO_WRITE_* instruction.

Workaround

As the host is actively causing the abort, it should be fully aware of the situation. The workaround now simply is to rewrite the IOADDR register (using the IO_SET_ADDRESS instruction) after each aborted block transfer.

Note: This usually is done anyway at the beginning of the next transaction.

OCDS_TC.H002 Setting IOSR.CRSYNC during Application Reset

If the host is shifting in a Communication Mode IO_READ_WORD instruction in the very moment an Application Reset happens, the read request flag (CBS_IOSR.CRSYNC) may be already set after the execution of the startup software. A monitor program may be confused by this and drop out of the higher

level communication protocol, especially if the host posts an instruction (with the IO_WRITE_WORD instruction) after detecting the reset.

Workaround

Two correlated activities should be incorporated in the tool software:

- After each reset the host should explicitly use CBS_IOCONF.COM_RST to reset any erroneously pending requests.
- The higher level protocol should require a specific answer to the very first command sent from the host to the device. Erroneous read requests then can be detected and skipped.

OCDS_TC.H003 Application Reset during host communication

Not only the host is able to cause resets of the device: External pins driven by the application, the internal watchdog and even the application program itself can trigger the reset generation process.

The only way to communicate reset events to the host is for Cerberus to reject the next instruction with “never-ending busy”, which should lead to communication time out on the host side.

The decision to accept or reject an instruction is done very early in the bit stream of the instruction. If an Application Reset happens after this point of time, the instruction will complete in most cases, and only the next one will be rejected.

As the temporal distance from reset event and instruction rejection is not fixed (apart from being sequential), it is highly recommended to check the IOINFO register (using the IO_SUPERVISOR instruction) each time an abnormally long busy period is experienced by the host. Especially a repetition of the rejected instruction should only be attempted if the possibility of Cerberus being in Error State has been excluded.

Workaround

Use IO_SUPERVISOR whenever a (too) long busy bit is observed.

OCDS_TC.H007 Early Acknowledgement of Channel Suspend for Active DMA Channel

This application hint is only relevant if a DMA channel is suspended for debug purposes and this DMA channel accesses a peripheral which is being suspended as well by the same event. If the DMA channel accesses memory, or if the accessed peripheral can be read/written without restrictions also in suspended state, there is no issue with the following behaviour.

When a debug suspend request is made while a DMA transfer is in progress, the specified behaviour is that the suspend request is only acknowledged on completion of the current DMA transfer.

However, in a specific corner case, if the suspend request is made in the same clock cycle as the start of the lowest priority channel with a pending access, i.e. the last DMA transfer to be executed, then the suspend request will be acknowledged immediately but the DMA transfer will continue to execute DMA moves until the current DMA transfer is complete.

Recommendation

Take into account that a peripheral may be suspended before the associated DMA triggering channel has entered the suspend state. Since this is a corner case which will rarely happen, please restart the debug situation if there are any indicators that this corner case has occurred.

SCU_TC.H004 Flag TRAPSTAT.SYSVCOLCKT after Power-on or System Reset

After a power-on (PORST) or System Reset, the PLL VCO loss-of-lock trap request flag TRAPSTAT.SYSVCOLCKT is set. As the PLL is not initialized by software at this point in time, the status of this flag is insignificant.

Therefore, flag TRAPSTAT.SYSVCOLCKT should be cleared by setting bit TRAPCLR.SYSVCOLCKT. Otherwise, an NMI trap will be generated when the corresponding source is enabled in register TRAPDIS.

SENT_TC.H001 Functionality of RCRx.CRZ: Differences to 2010 Standard

The SENT module has been developed according to the primary standard J2716-2008 published by SAE. Therefore, in standard frame format, bit RCRx.CRZ only controls the CRC augmentation of the fast channel (message CRC), but not of the (slow) serial message channel CRC.

In the J2716-2010 standard, augmentation of the 4-bit CRC (in standard frame format) is applied to both the (slow) serial message channel CRC and the fast channel (message CRC).

Recommendation

Implement CRC for (slow) serial message in software (see standard publication).

Note: Bit RCRx.CRZ does not control the augmentation for the extended serial messages. Extended serial messages are defined with a 6-bit CRC that always uses augmentation (independent of RCRx.CRZ).

SSC_AI.H002 Transmit Buffer Update in Master Mode during Trailing or Inactive Delay Phase

When the Transmit Buffer register TB is written in master mode after a previous transmission has been completed, the start of the next transmission (generation of SCLK pulses) may be delayed in the worst case by up to 6 SCLK cycles (bit times) under the following conditions:

- a trailing delay (SSOTC.TRAIL) > 0 and/or an inactive delay (SSOTC.INACT) > 0 is configured
- the Transmit Buffer is written in the last module clock cycle (f_{SSC} or f_{CLC}) of the inactive delay phase (if INACT > 0), or of the trailing delay phase (if INACT = 0).

No extended leading delay will occur when both TRAIL = 0 and INACT = 0.

This behaviour has no functional impact on data transmission, neither on master nor slave side, only the data throughput (determined by the master) may be slightly reduced.

To avoid the extended leading delay, it is recommended to update the Transmit Buffer after the transmit interrupt has been generated (i.e. after the first SCLK phase), and before the end of the trailing or inactive delay, respectively.

Alternatively, bit BSY may be polled, and the Transmit Buffer may be written after a waiting time corresponding to 1 SCLK cycle after BSY has returned to 0_B.

After reset, the Transmit Buffer may be written at any time.

SSC_AI.H003 Transmit Buffer Update in Slave Mode during Transmission

After reset, data written to the Transmit Buffer register TB are directly copied into the shift register. Further data written to TB are stored in the Transmit Buffer while the shift register is not yet empty, i.e. transmission has not yet started or is in progress.

If the Transmit Buffer is written in slave mode during the first phase of the shift clock SCLK supplied by the master, the contents of the shift register are overwritten with the data written to TB, and the first bit currently transmitted on line MRST may be corrupted. No Transmit Error is detected in this case.

It is therefore recommended to update the Transmit Buffer in slave mode after the transmit interrupt (TIR) has been generated (i.e. after the first SCLK phase).

After reset, the Transmit Buffer may be written at any time.

SSC_TC.H003 Handling of Flag STAT.BSY in Master Mode

In register STAT of the High-Speed Synchronous Serial Interface (SSC), some flags have been made available that reflect module status information (e.g. error, busy) closely coupled to internal state transitions. In particular, flag STAT.BSY will change twice during data transmission: from 0_B to 1_B at the start, and from 1_B to 0_B at the end of a transmission. This requires some special considerations e.g. when polling for the end of a transmission:

In master mode, when register TB has been written while no transfer was in progress, flag STAT.BSY is set to 1_B after a constant delay of 5 FPI bus clock cycles. When software is polling STAT.BSY after TB was written, and it finds

that STAT.BSY = 0_B, this may have two different meanings: either the transfer has not yet started, or it is already completed.

Recommendations

In order to poll for the end of an SSC transfer, the following alternative methods may be used:

- either test flag RSRC.SRR (receive interrupt request flag) instead of STAT.BSY
- or use a software semaphore that is set when TB is written, and which is cleared e.g. in the SSC receive interrupt service routine.

5 Documentation Updates

EBU_TC.D001 Documentation Updates related to SRI External Bus Unit (EBU)

1. Burst Phase Length

See TC1798 User's Manual V1.2 2012-05, Figure 15-30 and following:

A Burst Phase is always at least one clock cycle in length. The length of each Burst Phase (i.e. the number of EBU_CLK cycles) is derived from the value of the EXTCLOCK and EXTDATA fields in the EBU_BUSAPx registers. The length of the Burst Phase will either be:

- one period of BFCLKO if EXTDATA is 00_B
- two periods of BFCLKO if EXTDATA is 01_B
- four periods of BFCLKO if EXTDATA is 10_B
- eight periods of BFCLKO if EXTDATA is 11_B

The ratio of EBU_CLK to BFCLKO cycles is determined by bit field EXTCLOCK and can be either 1:1, 2:1, 3.1, or 4:1.

The length of the Burst Phase shown in Figure 15-30 and following is an integer number of BFCLKO periods which is determined by the EXTDATA field. It can actually be up to 32 (instead of 0..15) EBU_CLK cycles.

2. Command Delay and Address Phase Length

See TC1798 User's Manual V1.2 2012-05, Registers BUSRAPx/BUSWAPx:

The length (number of EBU_CLK cycles) of the Command Delay and Address Phase is programmed via bit fields CMDDELAY and ADDRC in registers EBU_BUSAPx, respectively. This makes it possible to select between 0..15 EBU_CLK cycles for the Command Delay Phase, and between 1..15 EBU_CLK cycles for the Address Phase.

3. Asynchronous Write Timings, Data Output Delay to WR#

See TC1793 Data Sheet V1.1.1 2012-12 / TC1798 Data Sheet V1.0 2012-03, Symbol t37:

The data output delay refers to the falling edge of WR#. Therefore, this parameter describes the “Data output delay to WR# falling edge, deviation from the ideal programmed value” (instead of WR# rising edge), as shown in the corresponding figures.

SSC_TC.D001 Documentation Updates related to SSC MRST/MTSR Master/Slave Attribute

This documentation update refers to column “Pin Functionality” of Table 10-18 (Port 4 Function), Table 10-24 (Port 7 Functions), and Table 10-24 (Port 18 Functions) of the TC1798 User’s Manual V1.2.

The attribute in parenthesis (“Master Mode” or “Slave Mode”) of some of the MRST/MTSR signals is not documented correctly.

An update is shown in the following **Table 8**.

Table 8 Update to SSC MRST/MTSR Master/Slave Attribute

Port Pin	I/O	Pin Functionality	Associated I/O Line
P4.0	I	SSC2 Input (Master Mode)	MRST2A
	O	SSC2 Output (Slave Mode)	MRST2
P4.1	I	SSC2 Input (Slave Mode)	MTSR2A
	O	SSC2 Output (Master Mode)	MTSR2
P7.0	I	SSC3 Input (Master Mode)	MRST3
	O	SSC3 Output (Slave Mode)	MRST3
P7.1	I	SSC3 Input (Slave Mode)	MTSR3
	O	SSC3 Output (Master Mode)	MTSR3
P18.0	I	SSC2 Input (Master Mode)	MRST2B
	O	SSC2 Output (Slave Mode)	MRST2
P18.1	I	SSC2 Input B (Slave Mode)	MTSR2B
	O	SSC2 Output (Master Mode)	MTSR2